Challenges for scheduling at the Edge

Denis Trystram, Giorgio Lucarelli, Clement Mommessin, Yanik Ngoko

Challenges for scheduling at the Edge

Denis Trystram
Giorgio Lucarelli - Clément Mommessin - Yanik Ngoko*

Univ. Grenoble Alpes, France
*QarnotComputing, Montrouge, France
Workshop on Mathematical Challenges on Scheduling, Sanya

Oct. 9, 2019

More than an Evolution...

Today, the digital world is *in action*:

The number of digital objects connected to Internet is growing exponentially.
Cyber Physical Systems are in interaction to each other and with humans.

...a Revolution.

Parallel and distributed computing platforms turn towards the *Edge*.
Compute everywhere and anywhen.

Sketch of the landscape

Hierarchical computing systems (4 levels)

1. HPC and Data centers (clouds)
2. Fog – small clusters (hundreds/thousands cores)
3. Edge – laptops/smart phones
4. IoT – sensors

A good example of points 3 and 4 is connecting flights at the airport:
Interaction between public and private transports, getting a lot of (local) informations, managing critical situations, etc..

Sketch of the landscape

Hierarchical computing systems (4 levels)

1. HPC and Data centers (clouds)
2. Fog – small clusters (hundreds/thousands cores)
3. Edge – laptops/smart phones
4. IoT – sensors

A good example of points 3 and 4 is connecting flights at the airport:
Interaction between public and private transports, getting a lot of (local) informations, managing critical situations, etc..

Challenges

Manage efficiently such complex infrastructures composed of multiple and heterogeneous digital/computing components.

- Large amount of polymorphic data issued from multiple sources.
- Resources appear and disappear.
- New storage capabilities.
- Heterogeneity characteristics of the different architectures, including OS kernels, compilers, etc.
- Network of different types and instability (different latencies).

WHERE running each workflow component?

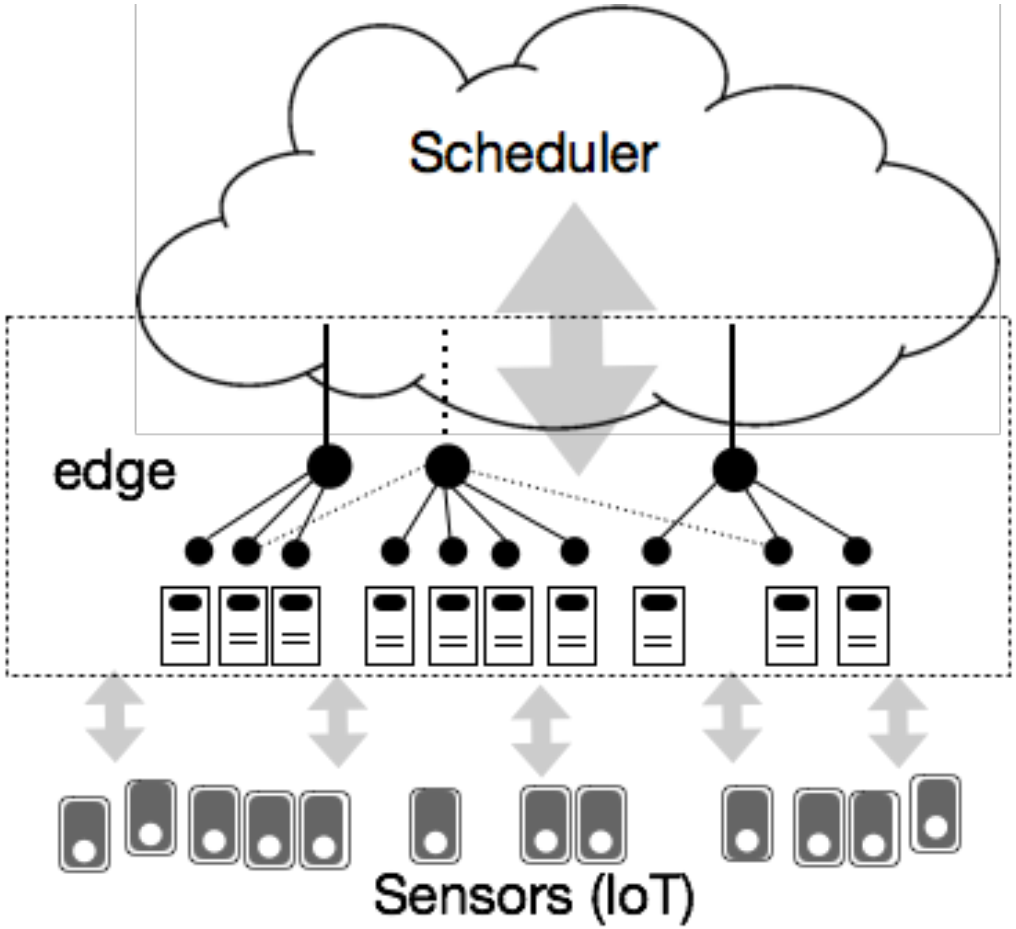
Challenges synthesis

Some issues/questions:

Compute as close as possible to the sensors/digital components to avoid data transfers (and also for privacy).

Need of new data storage abstraction: object versus file systems
data reduction/lossy compression.
Task flow versus data flow: **data streaming**.

Pictorally



Content of the talk

- Report a 3 years project (french nat. Research Agency) with the Qarnot Company

  Agence Nationale de la Recherche
  ANR
- Describe the problems
- Show and discuss preliminary scheduling solutions
- Simulation tools[1]

[1] Ideally, build a digital twin of the platform

Edge platform: case study at Qarnot

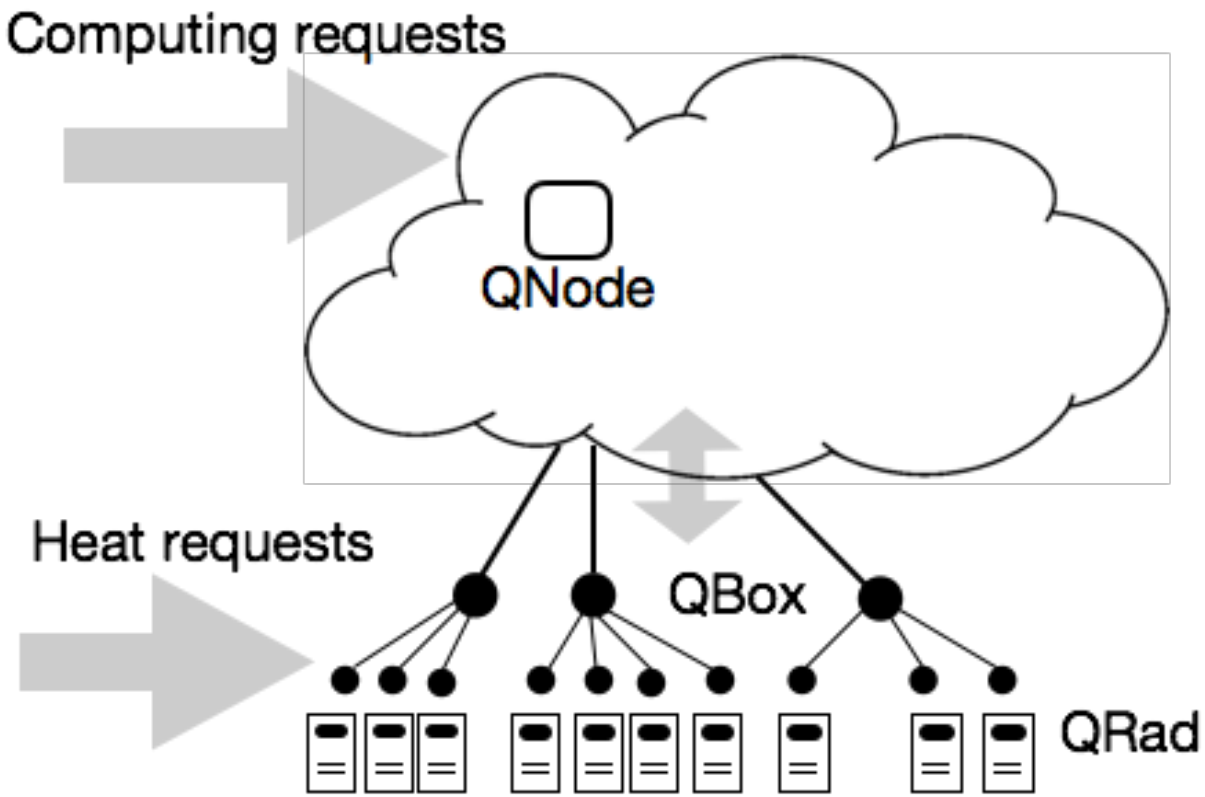
From Heaters to Computing Resources: **"turn IT waste heat into a viable heating solution for buildings."**

The actual Qarnot platform (Bordeaux and Paris):

- ∼1,000 distributed QRads embedding ∼3,000 diskless computing units
- ∼20 local servers (QBoxes) with memory disks
- 1 global server (QNode) with a centralized storage server

Credits: `https://www.qarnot.com`

Qarnot Platform



Two types of computing requests

Cloud tasks:

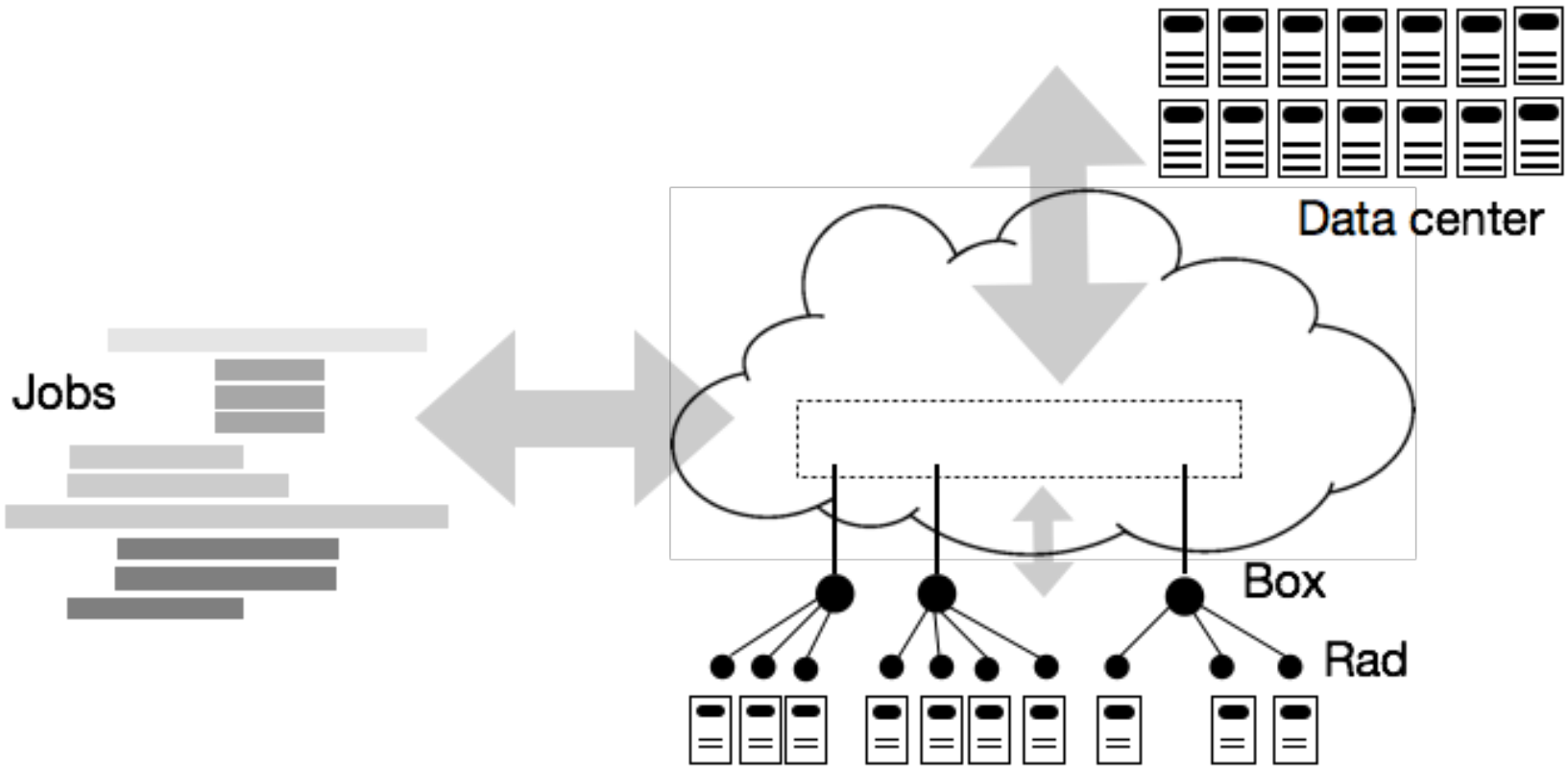
- Submitted to the QNode
- Have data-set dependencies in the centralized storage
- Have different priorities (low or high)

IoT tasks:

- Submitted to a – local – QBox
- Have data-set dependencies in the QBox disk
- Have different priorities (low, high or very high)
- Should be executed **locally**

Abstract view of the Qarnot Platform

Tasks/jobs (groups of **sequential instances**) are submitted on-line.



Target of the project

Main goal: design, implement and test different placement and scheduling policies at both QNode- and QBox-levels.

Involved People:

- A PhD student
- A PostDoc (18 months)
- A engineer (12 months)

Platform Dynamicity

Resources appear and disappear over time: **the inhabitants decide according to the weather or their schedule!**

- Available resources when heating is required (QRad is ON)
- Unavailable when ambient air is too warm (QRad is OFF)

$\rightarrow$ Also depends on the task priority

Network uncertainties:

- Link failures
- Congestion/contention

Two-Level Scheduling Problem

Remind: **The only computing power is in QRads**

Make **global decisions** at QNode-level:

- Decide where to dispatch (groups of) **instances**
- Ensure global load-balancing

Make **local decisions** at QBox-level:

- Schedule instances on QRads
- Regulate room temperature (via DVFS – frequency scaling)
- Ensure heating needs are satisfied

Need to reach 100% of platform usage.
$\rightarrow$ **An idle QRad is a lack of heating**

Multiple Objectives problem

Various objectives for different actors:

- Cloud users: Minimize waiting/completion time of tasks
- IoT tasks: Responsiveness
- Inhabitants: Minimize the gap to target temperature.
- Qarnot:
  Maximize tasks throughput (profit)
  Minimize lack of heating

Qarnot Solution: naive on-line

Periodic reports (∼30 sec.) from QBox to QNode with:

- Number of resources available for each task priority
- Amount of free space on disk

QNode-scheduling:

- Sort QBoxes by least available resources first (no temperature knowledge)
- Sort tasks by highest priority first
- For each task, dispatch as many instances as possible

QBox-scheduling:

- Retrieve data-set dependencies
- Schedule high priority instances on coolest QRads
- Schedule low priority instances on warmest QRads

Temperature Regulation

Frequency and temperature regulator in each QRad:

- In general: speed scaling (DVFS) on each multi-core to adapt power consumption
- When too warm: instances are killed and re-submitted to the QNode
- When too cold (lack of heating): *"background"* compute-intensive instances are generated (best-effort blockchain mining[2])

[2] Open problem: it could be improved

Global scheduler

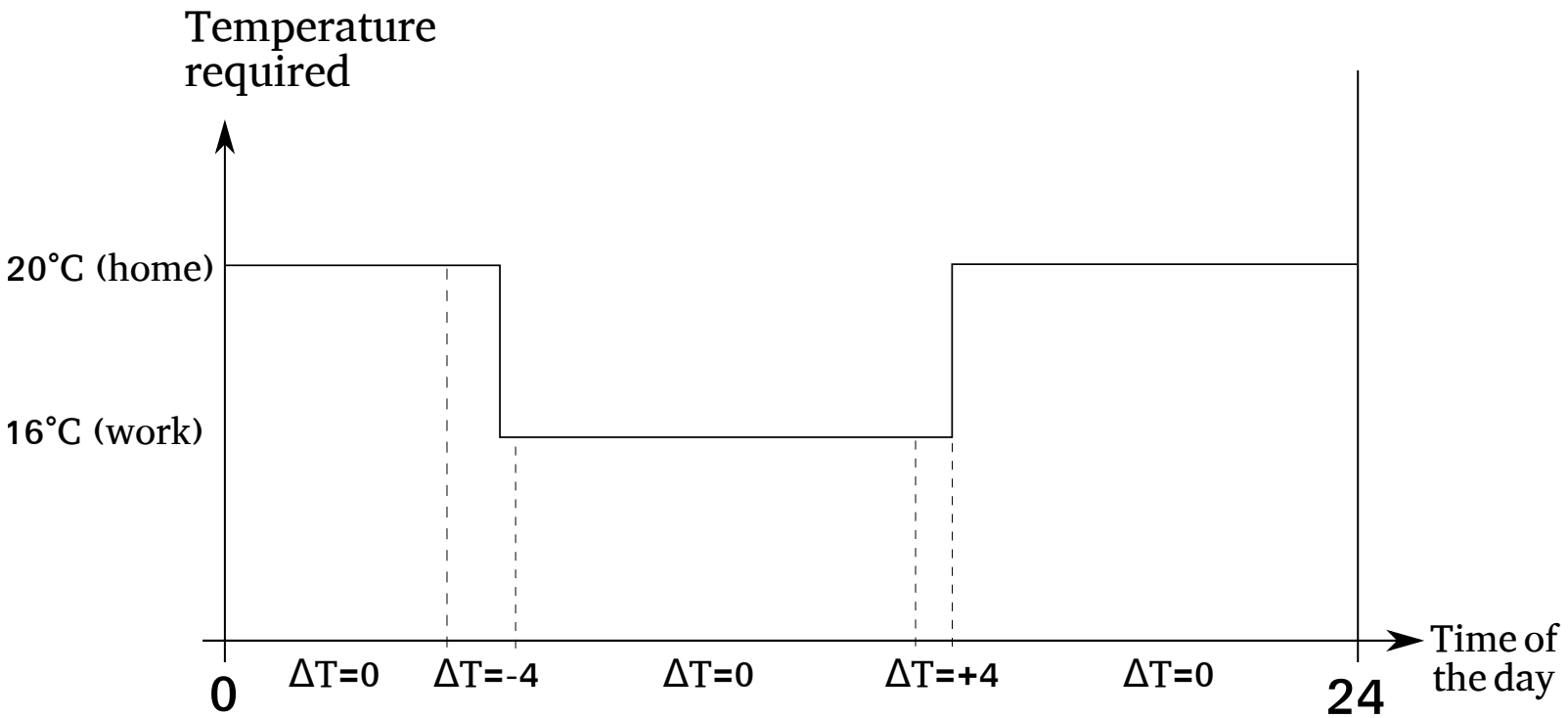
Several policies for the QNode dispatcher:

- Standard
- Locality-Based
- Replicate 3 times data set on the Least Loaded Disk
- Replicate 10 times (again with LLD)
- FullReplicate (allows instantaneous transfers)

Implementation with standard Qarnot scheduler at QBox-level.

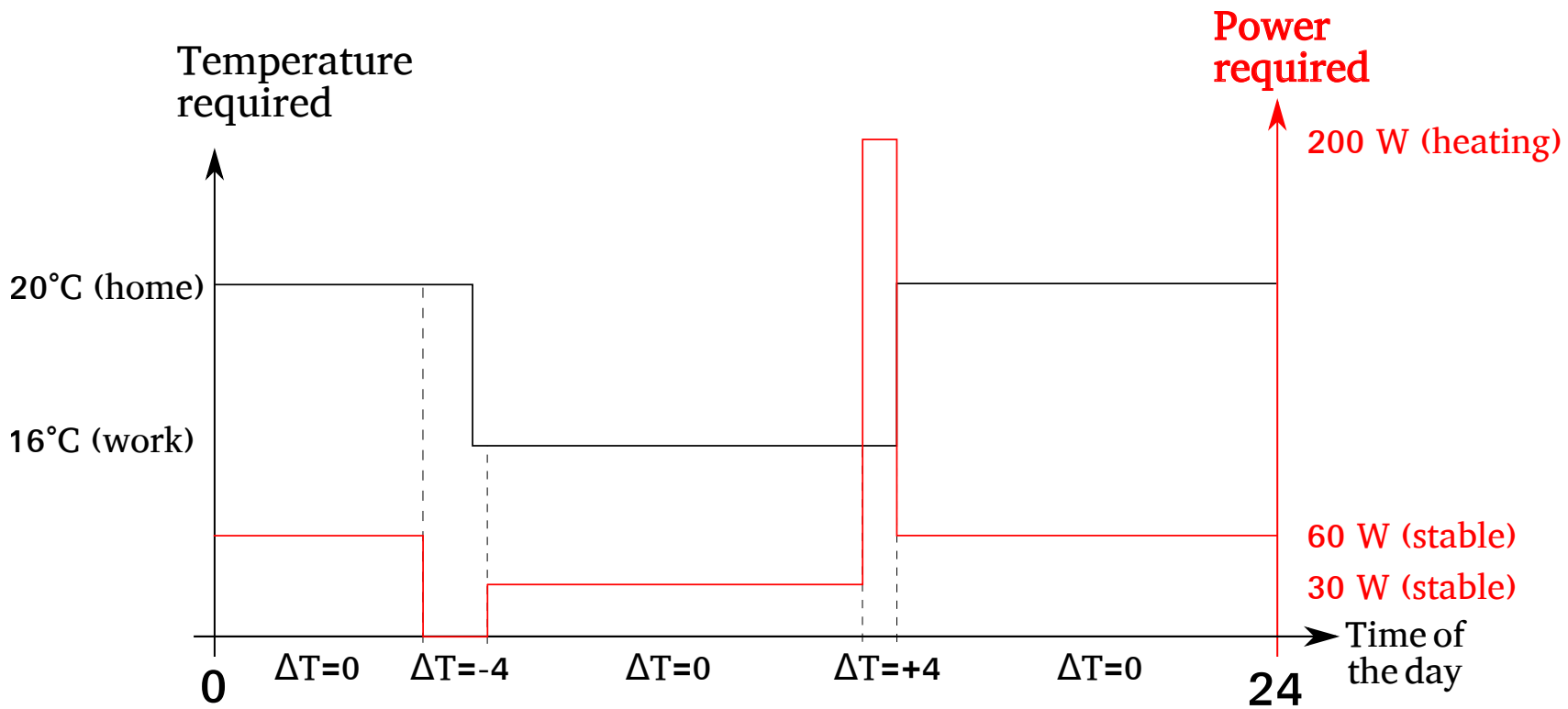
For the moment, no local tasks are considered.

QBox-level Scheduling



Target temperature diagram

QBox-level Scheduling



Power diagram

Main problem: testing on the production platform is not conceivable (and it would take time...).

$\rightarrow$ **Simulation is needed!**

Remark: the theoretical analysis is difficult.

Main problem: testing on the production platform is not conceivable (and it would take time...).

$\rightarrow$ **Simulation is needed!**

Remark: the theoretical analysis is difficult.

Simulation Framework

SimGrid[3]: Large-scale distributed system simulator with execution and communication models.
→ Used to simulate platform (including temperature) and tasks execution.

Batsim[4]: Infrastructure simulator for jobs and I/O scheduling.
→ Used to drive the simulation, submit tasks and communicate with the decision process.

Pybatsim[5]: Batsim's Python API exposing methods to easily communicate with the Batsim process.
→ Used to implement both QNode and QBox schedulers.

[3] `https://github.com/simgrid/simgrid`
[4] `https://gitlab.inria.fr/batsim/batsim/tree/temperature`
[5] `https://gitlab.inria.fr/batsim/pybatsim/tree/temperature`

Implementation

Dedicated module for convert computations to temperature.

2 solutions have been developed:

- Learning on actual logs (problems: old generation of Qrads and missing contextual informations).
- Analytic expression (given by physicians).

Remark: better than what is available today at Qarnot (empirical law which causes a lot of instabilities).

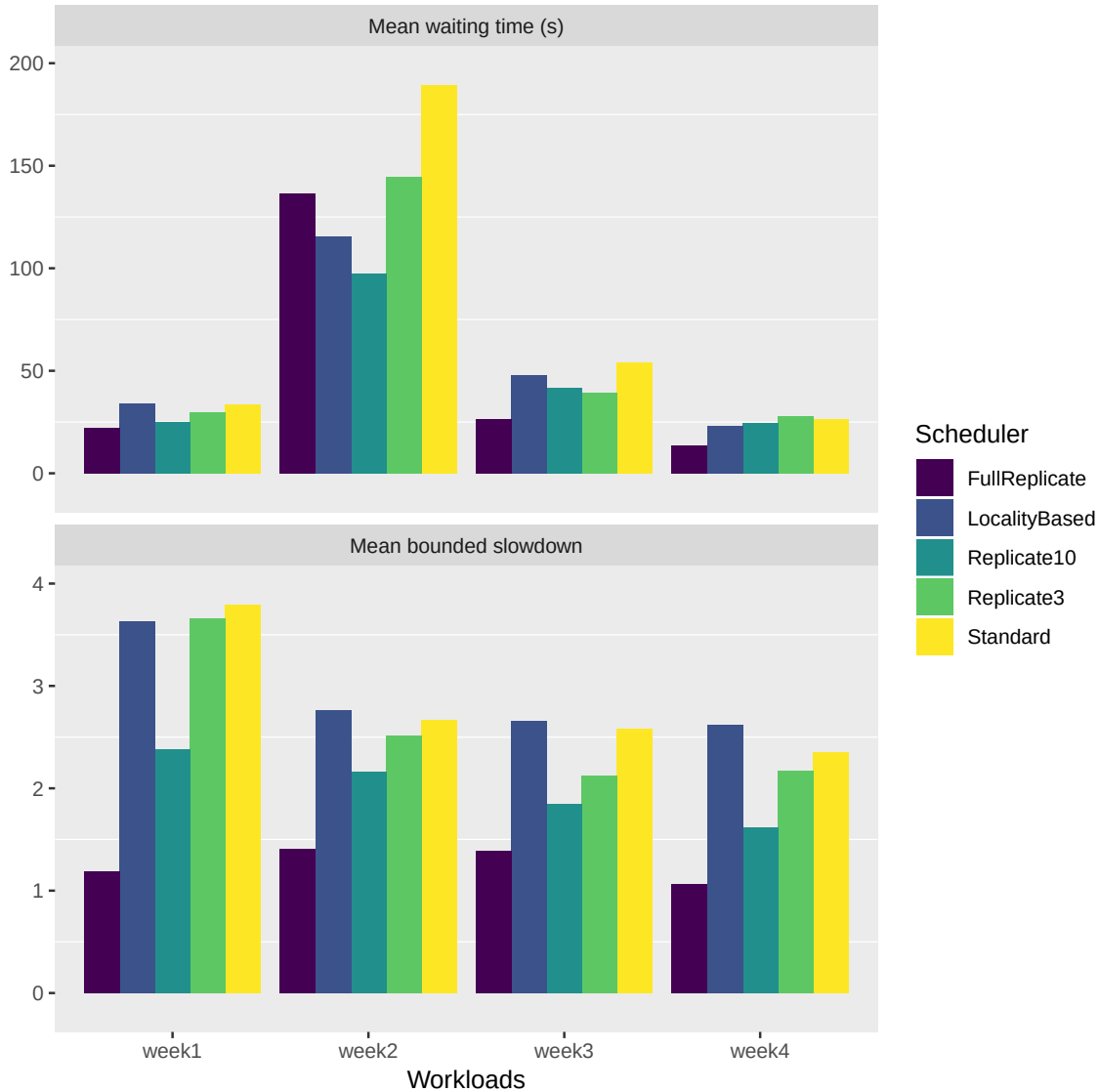
Simulations

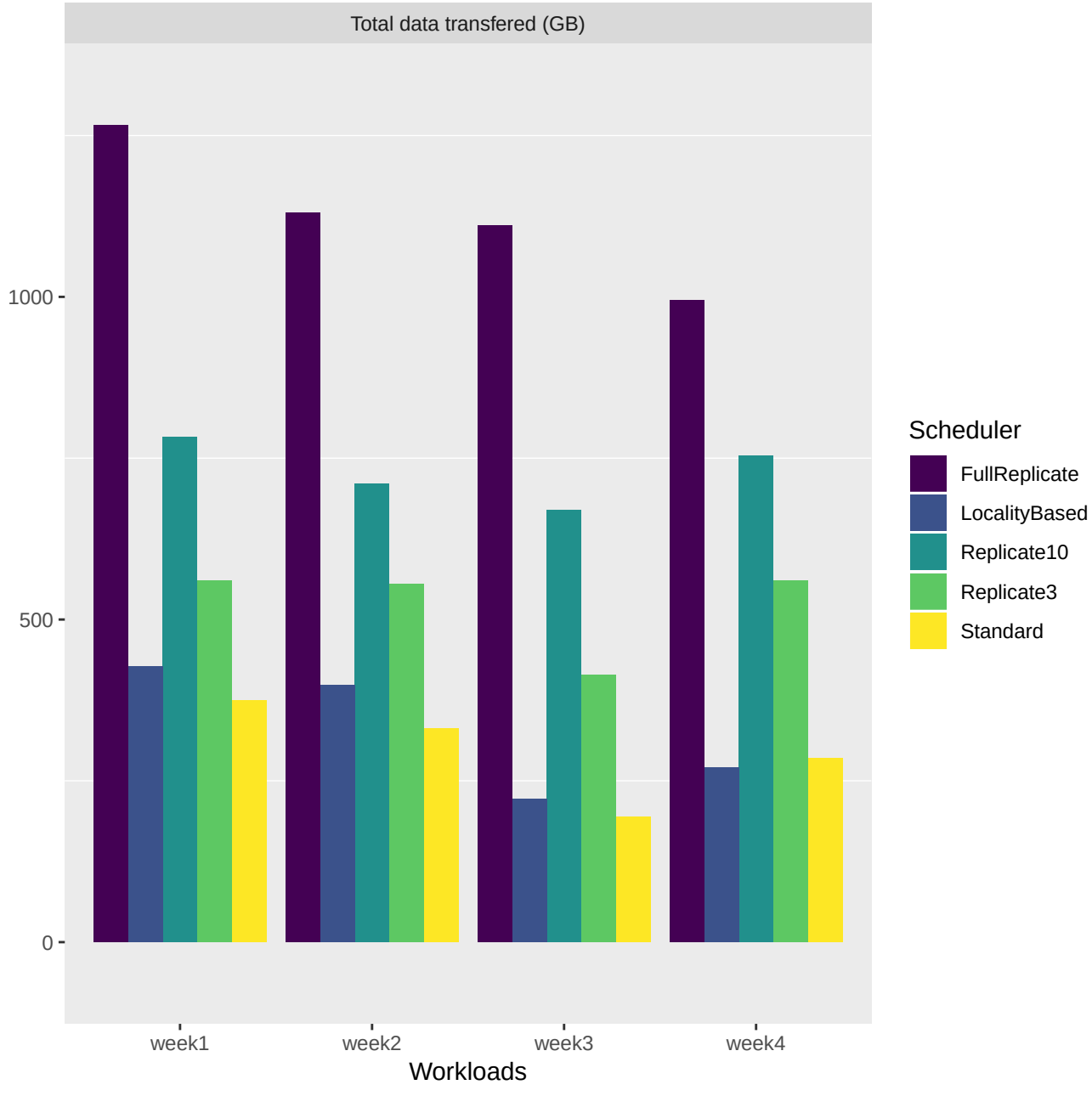
Run 4 times the logs on a week (between may 3 and june 1, 2019).

The simulated platform was composed of 3390 QMobos of 669 QRads, managed by 20 QBoxes.

Two metrics studied:

- mean waiting time
- bounded slowdown (normalized average stretch)


Mean waiting time (s)
200
150
100
50
0
Mean bounded slowdown
4
3
2
1
0
week1
week2
week3
week4
Workloads
Scheduler
FullReplicate
LocalityBased
Replicate10
Replicate3
Standard


Total data transfered (GB)
1000
500
0
week1
week2
week3
week4
Workloads
Scheduler
FullReplicate
LocalityBased
Replicate10
Replicate3
Standard

Local scheduler (model)

Two sets of tasks:

- $\mathscr{S}_1$ global tasks.
  off-line (sequential) tasks with their release dates (corresponding to data transfers).
- $\mathscr{S}_2$ local tasks.
  Sequential tasks arriving on-line.
  Not too many (10 to 20 %), usually small ones.

Qarnot platform is rather homogeneous (uniform machines).

Notice here that rejection (interrupt a task and re-execute it elsewhere) makes sense.

Local scheduler (Objectives)

Two directions are under investigation.

- **Single objective.**
  Minimize max flow time for $\mathscr{S}_2$ under the capping on sum flow for $\mathscr{S}_1$.
- **Bi-objective.**
  Minimize average flow time or tardiness for $\mathscr{S}_1$ and minimize max flow time for $\mathscr{S}_2$

Some results

Qarnot platform.

- Preliminary results assuming periodic arrivals in $\mathscr{S}_2$:
  Study the problem for one machine, then extend.
- Interface between global and local schedulers: add a module which gives the pressure of local tasks.
- Learning algorithms have been developed (get a better idea of the leaving habits).

Extend to more general edge platforms.

Conclusion

General message: It is not so easy to build a bridge between theory and practical tools.

- Model of the actual Qarnot/edge platform.
- Simulator (almost complete).
- We proposed several scheduling problems .
  Some are still under investigation
  Theoretical analysis remains to be done
- A preliminary solution has been implemented and assessed on real execution traces.